Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L16	L15 and (updat$ or load$ or download$)	321	L16
L15	L14 and provid$ near5 (configuration$ or operating system or prerequisit$)	368	L15
L14	check$ near4 (configuration$ or operating system or prerequisit$)	3726	L14
L13	check$ nea4 (configuration$ or operating system or prerequisit$)	0	L13
L12	L11 and l6	12	L12
L11	provid$ near5 (prerequisit$ or (appropriat$ near5 configuration$))	327	L11
L10	L9 and provid$ near5 (config$ or prerequisit$ or software or hardware)	20	L10
L9	l6 and (instal$ or updat$ or load$ or download$)	214	L9
L8	L6 and (provid$ near5 prerequisit$)	12	L8
L7	L6 and (provid$ near5 prereuisit$)	0	L7
L6	(check$ or determin$) near4 prerequisit$	339	L6
L5	L4 and (operating system$ or os$)	1	L5
L4	6151643.pn.	1	L4
L3	L2 and (without near4 user$)	25	L3
L2	L1 and software and hardware	100	L2
L1	automatic$ near5 determi$ near5 configuration$	236	L1

END OF SEARCH HISTORY

DOCUMENT-IDENTIFIER: US 6496858 B1
TITLE: Remote reconfiguration of a secure network interface

Detailed Description Text (9):
The remote server 206 represents central facility for providing convenient and efficient configuration and maintenance of the gateway interface device. In one embodiment of the present invention, the remote server 206 (hereinafter referred to as the "remote management server") is connected to ISP 204 and maintains a dynamic dialog with ISP 204 to configure and maintain gateway interface device 208 in client network 220. Remote management server 206 interacts with gateway interface device 208 to provide configuration information and upgrade parameters required by the gateway interface device 208. In this manner, remote management server 206 basically serves as a repository for information required by the gateway interface device 208. Such information may include configuration information related to LAN 210, internet address blocks, internet domain names, and data related to the physical and logical interfaces between the client network 220 and ISP 204.

Detailed Description Text (11):
Remote management server 206 and gateway interface device 208 contain security information such as passwords and encryption keys that are used to establish a trust relation sufficient to ensure secure remote configuration and upgrade of gateway interface device 208. By providing a configuration management function within remote management server 206 which is registered with an ISP 204, it is possible to download configuration and upgrade information and parameters to gateway interface device 208 at the time the gateway interface is first installed between the client network 220 and the telephone client 204. This eliminates the requirement that the network administrator program the network interface device with such configuration and initialization information. This system thus greatly reduces the amount of work required to connect client network 220 to an internet.

Detailed Description Text (27):
In step 618, the configuration manager checks whether there are further requests to be included in the transaction. If further requests are to be processed, the process proceeds from step 606 and the user inputs a further service request through the user interface. If, in step 618, no further requests are determined to be included, the user requests the transaction to be committed, step 620. The configuration manager then propagates the commit request to each applicable service manager, step 622.

system using one or more interfaces; instructions for implementing the one or more hardware objects in the reconfigurable computing circuitry; and instructions for storing a plurality of system information in the memory circuitry; wherein the reconfigurable computing circuitry further comprises at least one Field Programmable Gate Array ("FPGA") providing reconfigurable logic for the one or more configuration files to define the one or more hardware objects; and wherein said one or more configuration files further comprise a plurality of versions of each of the one or more configuration files, wherein each configuration file in said plurality of versions provides the same or similar functionality and is targeted for configuring a different type of the at least one FPGA.

L15: Entry 8 of 368 File: USPT Mar 25, 2003

DOCUMENT-IDENTIFIER: US 6539438 B1
TITLE: Reconfigurable computing system and method and apparatus employing same

Brief Summary Text (20):
Moreover, there is a need for a partitionable reconfigurable computing system with the ability to allow hardware objects to be partitioned into the same or different FPGAs on the same board and allocated intelligently as the need for the hardware objects arises. There is also a need for a reconfigurable computing system that allows for multiple configuration files providing the same or similar functionality that are each targeted and configured to properly interface with different types of FPGAs (i.e., difference in either gate capacity, size, vendor, etc.)

Detailed Description Text (43):
FIG. 5 is a flowchart of the Quick Qard technology application software set-up procedure showing one process that can be used to set up application software programs for use with Quick Qards 26 of FIGS. 2 and 4. At step 86 an application software program is started (spawned). At step 88, the application software program communicates with QQT software driver 10 and presents a list of hardware objects for the application to use. The application software program may also present the name of a configuration file or files that it recommends QQT software driver 10 use. At step 90, QQT software driver 10 checks if the requested configuration file(s) and hardware object(s) are already available for use in the system. If they are, at step 100 QQT software driver 10 uses them by setting up or up-dating the sharing control for each of the hardware objects and IRQs, if needed. At step 102, QQT software driver 10 returns a handle to the application program for each of the requested hardware objects. Set-up is complete at step 104.

CLAIMS:

12. A reconfigurable computing method for interfacing one or more application programs running on a host system to one or more hardware objects defined in one or more configuration files, comprising: initializing a reconfigurable computing software driver for managing, configuring and reconfiguring reconfigurable computing circuitry comprising flexibly configurable circuitry using the one or more configuration files; communicatively connecting the reconfigurable computing circuitry and memory circuitry associated with the reconfigurable computing circuitry to the host system using one or more interfaces; implementing the one or more hardware objects in the reconfigurable computing circuitry; storing a plurality of system information in the memory circuitry; and interfacing the one or more hardware objects with at least one of the one or more application programs using the reconfigurable computing software driver; wherein the reconfigurable computing circuitry further comprises at least one Field Programmable Gate Array ("FPGA") providing reconfigurable logic for the one or more configuration files to define the one or more hardware objects; and wherein said one or more configuration files further comprise a plurality of versions of each of the one or more configuration files, and wherein each configuration file in said plurality of versions provides the same or similar functionality and is targeted for configuring a different type of the at least one FPGA.

24. A reconfigurable computing system for interfacing one or more application programs running on a host system to one or more hardware objects defined in one or more configuration files, comprising: a reconfigurable computing software driver for managing, configuring and reconfiguring reconfigurable computing circuitry comprising flexibly configurable circuitry using the one or more configuration files; instructions for interfacing the reconfigurable computing circuitry and memory circuitry associated with the reconfigurable computing circuitry to the host

L15: Entry 48 of 368 File: USPT May 21, 2002

DOCUMENT-IDENTIFIER: US 6393557 B1
TITLE: Dynamic method for configuring a computer system

Detailed Description Text (13):
FIG. 3 is a flowchart of a method for configuring a computer system, using device configuration information such as shown in FIG. 2. The method at system initialization begins by creating a root object and scheduling it for probing (24). Creating a root object includes examining the device configuration information for a "root" device in the system. Its device configuration file is examined to determine its device type and interface types, and the root object is then created. The root object is then pulled from the probe list as the current object for probing (26). The outgoing interface types the current object provides are determined (28). The device configuration files (FIG. 2) are then checked for device types that can exist on the outgoing interface types provided by the current object (30). By "exist" is meant a device type that has an incoming interface type that matches an outgoing interface type of the current object and hence can connect to the device associated with the current object. Devices of a type that can exist on an outgoing interface of the current object are then probed for (30).

Detailed Description Text (15):
Applying this part of the method to the computer system of FIG. 1, system interconnect 20 is the root device, and a root object is created for it. The system interconnect object is scheduled for probing and then, as the only element on the probe list, is pulled from the list as the current object. The only outgoing interface type for the system interconnect object is determined to be the SCI interface type. The device configuration files are then checked for device types that have an incoming SCI interface type. The only device type having an incoming SCI interface type is the node device type. The SCI ports and units of the system interconnect object are then probed for the presence of nodes.

CLAIMS:

1. In a computer system, a method for configuring devices within the system comprising:

providing device configuration information for types of devices that may be included within the computer system, the device configuration information for a device type including interface types by which devices of that device type connect to other devices;

determining the interface types of a current object associated with a device;

for each device type that can exist on a determined interface type of the current object, probing for devices of that type;

creating a child object for a device found from the probing and a link between the current object and the child object;

making a child object the current object; and

repeating the above actions at least until the devices to be configured within the system have associated objects linked together.

17. In a computer system, a method for configuring devices within the system comprising:

providing device configuration information for types of devices that may be included within the computer system, the device configuration information for a device type including interface types by which devices of that device type connect to other devices;

determining the interface types of a current object associated with a device;

for each device type that can exist on a determined interface type of the current object, probing for devices of that type;

for a found device that has no associated object, creating a child object for the found device and a link between the current object and the child object;

for a found device that has an associated object, creating a link between the current object and the child object;

making a child object the current object; and

repeating the above actions at least until the devices to be configured within the system have associated objects linked together.

End of Result Set

Generate Collection | Print

L3: Entry 25 of 25 File: USPT May 7, 1991

DOCUMENT-IDENTIFIER: US 5014193 A
TITLE: Dynamically configurable portable computer system

Abstract Text (1):
A dynamically configurable computer system which includes apparatus for storing former and sensed current system configuration data and for automatically reconfiguring the system without user interaction.

Brief Summary Text (7):
In the past, portable computer users were required to manually reconfigure the system each time a new environment of peripheral devices was encountered, including the situation where the the portable unit was being used in its self-contained configuration where no external peripherals were attached. Re-configuration typically involved setting switches or jumpers to indicate the presence or absence of external disk drives, whether such drives were lo be designated drive "A" or drive "B", whether a particular communication port was connected or not connected to a modem or serial printer, etc. Re-configuration also included execution of system initialization software or set up sequences to identify for the central processing unit the type or specification of disk drive which was connected as drive A or drive B and other system configuration data.

Brief Summary Text (9):
In contrast to such prior art systems, the dynamic configuration system of the present invention provides the portable computer with additional peripheral status or configuration storage registers and a novel software driven means for dynamically reconfiguring the system without user interaction. The system of the present invention includes means for determining the peripheral status and updating the configuration data to permit use of the computer to begin without executing a user prompted initialization sequence or requiring manual switches or jumpers to be reset each time the user changes environments.

Brief Summary Text (10):
The system includes a readable/writable memory device suitable for low power use, such as a CMOS memory circuit which stores data relating to peripheral configurations in a number of designated address locations or registers. The system of the present invention retains data relating to peripheral configuration including floppy disk drive type data when the system was last used utilized, automatically determines the current system configuration on power up, and updates appropriate data registers to reflect the current configuration status. By storing the data relating &o the last configuration status, the system is able to verify whether there has been a change, and if so update the configuration. If no change has occurred, the initialization sequence is by-passed and operation of the portable computer begins immediately. In this way, the user who typically alternates between two configurations, i.e. office and home, or office and self-contained while traveling will encounter little delay or inconvenience resulting from the changed configuration.

Detailed Description Text (2):
The computer system of the present invention provides a means for configuring or changing the configuration of the system without the utilization of switches or jumpers, eliminating the need for the user to access internal system hardware and without the need to run a special configuration routine each time the configuration is changed. Referring now to FIG. 1, the letter S designates generally a typical personal computer system suitable for use with the present invention. Computer

system S includes a central processing unit 10, a memory circuit 12 including system read only and random access memory circuits, a liquid crystal display 14, a serial communications port 16, a second serial communications port which generally includes the circuitry forming a modem 18, an internal fixed or hard disk memory device 20, an internal floppy disk drive 21, a printer port 22, a low power consumption readable, writable memory circuit 24, typically a complimentary metal oxide or CMOS device, an external expansion floppy-disk control logic circuit 26 for interaction with external floppy-disk drive unit 28, a keyboard and associated interface circuit 29 and a miscellaneous input/output device control logic circuit 30 for use with various external devices such as alternating current adapter 32.

Detailed Description Text (4):
Briefly stated, the system of the present invention utilizes a number of novel instruction sequences stored with the system's power up initialization commands to control the initialization of the computer system S, determine peripheral configuration and make appropriate changes without user interaction. These instruction sequences cause the processor 10 to read the status of peripheral devices and system variables, store configuration data relating to changes detected and update the system configuration without user interaction. The system S includes in memory 12 various cable locations which store system peripheral device configuration values. On power up, the software of the present invention clocks in data to a drive status latch for each drive to reflect system status on power up. The status of these latches can only be modified during the initialization sequence and consequently do not react to changes in configuration made after initialization while the system is operating. In order to facilitate dynamic configurability, a number of readable/writable memory locations are also provided in CMOS circuit 24 to store system configuration data. On power up, the system software causes a polling of present conditions to be made and data relating to that status is stored in appropriate data registers in CMOS circuit 24. A comparison between the current data in these configuration registers to that in system memory tables relating to the system's prior configuration is made to determine if changes are necessary to the configuration status. If so, the changes are read from the current data registers in CMOS circuit 24 to the appropriate memory tables in internal memory 12.

Detailed Description Text (12):
Referring now to the drawings, the sequence of instructions utilized by computer system S to dynamically configure the external drive status and communication ports will be described in detail. These instructions are executed on system power up to poll the status of the external drive and communications ports, determine if they are valid, and if so, automatically make the indicated changes to the configuration registers in system memory 12.

Detailed Description Text (23):
Where the external drive status bits comprise 01 11, a valid configuration change of an attached external drive from drive A to drive B is indicated. This status code causes control to be transferred from step 108 to subsequence 6 (FIG. 8). This situation can occur if the user goes from a first peripheral system at the office wherein the external drive was designated drive A to the home or other system where the external drive is attached but designated drive B or if the user changes the setting of the A/B switch on his external drive. The system of the present invention initially assumes the external drive will be of the same type and merely transfers the drive type data from one register to the other. If this assumption is incorrect, other portions of the basic system software will generate a setup error message and request the user to run the system setup program.

Detailed Description Text (26):
Where the external drive status bits comprise 11 01, a valid configuration change from an attached external drive designated B to drive A as indicated. In this situation, control is transferred from step 108 to subsequence 8 (FIG. 10). Subsequence 8 begins execution at step 166 which causes the processor to transfer the drive B type data temporarily to an internal memory location. Thereafter, step 168 causes the processor to transfer the drive A type data to the drive B type data register, and step 170 transfers the temporarily saved drive A type data to the drive B type data register from internal memory. Control is thereafter transferred to step 172 which causes the EDS register to be updated. Step 173 updates the CMOS checksum value, followed by continue step 174 which returns control for further initialization. In this situation, the system of the present invention initially assumes the drive type data for the external drive will be the same, and if not, the basic system software will generate an error message requiring set up values for the

external drive to be provided by the user.

CLAIMS:

1. In a computer system including a processor, system memory, a housing, provisions for connecting one or more peripheral devices such as disk drives, serial printers, or modems into the computer system internal or external to the housing and peripheral status data storage means for storing data relating to any connected peripheral devices, a dynamic configuration means for automatically initializing system peripheral configuration data on system power-up, said dynamic configuration means comprising:

(a) system configuration status means for storing data indicative of internal and external peripheral device status;

(b) former status means for storing data indicative of the status of internal and external peripheral devices when the system was last used;

(c) current status means for determining the current status of internal and external peripheral devices and developing data indicative thereof;

(d) means coupled to said former status means and said current status means for determining differences between said data of said former status means and said current status means; and

(e) means coupled to said difference determining means and said system configuration status means for automatically updating said data stored in said system configuration status means based upon said determined differences between said former status data and said current status data.

L8: Entry 4 of 12 File: USPT Aug 27, 2002

DOCUMENT-IDENTIFIER: US 6442754 B1
TITLE: System, method, and program for checking dependencies of installed software components during installation or uninstallation of software

Detailed Description Text (19):
Other functions of the tool kit include i) providing install property objects that contain variables as values that become defined for a specific operating environment; ii) enabling a property value to be temporarily overridden; iii) a software state machine that enables a programmer to easily customize an install program by merely adding, deleting, or changing the various states that contain the functions and flow of control of the program; iv) automatically detecting a programming error if a programmer incorrectly specifies a non-existent state within the state machine; v) automatically selecting a system-dependent function; vi) a containment structure consisting of program object, fileset objects, install objects, where each fileset object and install object contains means to install and uninstall itself and to log itself; vii) enabling the management of folders, shortcuts and icons; viii) enabling environment variables to be read, created, modified and deleted; ix) providing dependency checking of prerequisite programs during both install and uninstall; and x) providing various logs, e.g. a log for keeping track of what is being installed, and a log that reports the progress of install. Logs are used for both the install and uninstall process. Furthermore, these logs are human readable which allows them to be checked, e.g., after a silent install, to ensure that a file has installed successfully. The tool kit also enables multiple destination directories to be installed from multiple source directories. For example, there can be multiple components of file sets included in an install where a file set is a separately installable/uninstallable piece of code or set of files.

DOCUMENT-IDENTIFIER: US 6493594 B1
TITLE: System and method for improved software configuration and control management in multi-module systems

Abstract Text (1):
The system and method of the invention automatically provide appropriate configuration and control software to a hardware system consisting of multiple target hardware modules. A system controller is connected to the target hardware modules and to a repository that stores a set of software modules. Each software module includes configuration and control information for a particular target hardware module. When a particular software control and configuration scheme for at least a portion of the target hardware modules is to be implemented in the hardware system, software modules corresponding to the particular software control and configuration scheme are selected and associated with a system definition file that is also stored in the repository. When the system definition file is invoked by the system controller, the system controller executes the associated software modules thereby providing the particular software control and configuration scheme to appropriate target hardware modules. When the particular software control and configuration scheme needs to be updated, one or more software modules of the system definition file may be replaced with an updated version without reconfiguring the hardware system.

Brief Summary Text (3):
The present invention relates to a system and method for improving software configuration and control management in a hardware system. More particularly, the invention is directed to a system and method for automatically providing appropriate configuration and control software to a hardware system consisting of a controller and multiple target hardware modules connected thereto.

Brief Summary Text (7):
It would thus be desirable to provide a system and method for automatically providing appropriate configuration and control information to a multi-module hardware system when the hardware system is updated or the functionality of the hardware system is changed without replacement of the system software module. It would further be desirable to effectively manage changes in desirable configuration and control schemes for a multi-module hardware system.

CLAIMS:

1. A system for providing an appropriate software control and configuration scheme to a multi-module system having a system controller and a plurality of target hardware modules connected to the system controller, comprising: a plurality of software modules, each of said plurality of software modules comprising configuration and control information for a particular target hardware module from said plurality of target hardware modules; a software module editor for at least one of editing and creating at least one of said plurality of software modules; at least one system definition file representative of a particular software control and configuration scheme for the multi-module system, wherein the particular software control and configuration scheme comprises at least a portion of said plurality of software modules; a system definition file editor for at least one of editing and creating said at least one system definition file; selection means in the system controller for selecting one of said at least one system definition file, said selected system definition file corresponding to a desired particular configuration and control scheme; and repository means connected to the system controller for storing said plurality of software modules and said at least one system definition file; wherein the system controller is operable for executing said selected system

definition file by loading the software modules which comprise said desired particular configuration and control scheme into at least a portion of said plurality of target hardware modules; and wherein, when a new target hardware module is added to the multi-module system, a new software module comprising configuration and control information for the new target hardware module is created using the software module editor and at least one of: a new system definition file is created using the system definition file editor, where the new system definition file includes the new software module in the particular software control and configuration scheme of the new system definition file; and at least one of the at least one system definition file is edited to include the new software module in the particular software control and configuration scheme of the edited at least one system definition file.

DOCUMENT-IDENTIFIER: US 6442754 B1
TITLE: System, method, and program for checking dependencies of installed software components during installation or uninstallation of software

Detailed Description Text (19):
Other functions of the tool kit include i) providing install property objects that contain variables as values that become defined for a specific operating environment; ii) enabling a property value to be temporarily overridden; iii) a software state machine that enables a programmer to easily customize an install program by merely adding, deleting, or changing the various states that contain the functions and flow of control of the program; iv) automatically detecting a programming error if a programmer incorrectly specifies a non-existent state within the state machine; v) automatically selecting a system-dependent function; vi) a containment structure consisting of program object, fileset objects, install objects, where each fileset object and install object contains means to install and uninstall itself and to log itself; vii) enabling the management of folders, shortcuts and icons; viii) enabling environment variables to be read, created, modified and deleted; ix) providing dependency checking of prerequisite programs during both install and uninstall; and x) providing various logs, e.g. a log for keeping track of what is being installed, and a log that reports the progress of install. Logs are used for both the install and uninstall process. Furthermore, these logs are human readable which allows them to be checked, e.g., after a silent install, to ensure that a file has installed successfully. The tool kit also enables multiple destination directories to be installed from multiple source directories. For example, there can be multiple components of file sets included in an install where a file set is a separately installable/uninstallable piece of code or set of files.